Efficient Implementation of LPC Algorithm for Voice Decoding Process

Mohamed Atri, Fatma Sayadi, Rached Tourki

Electronics and Microelectronics Laboratory, Faculty of Sciences Monastir, Tunisia

Mohamed.Atri@fsm.rnu.tn; Sayadi_fatma@yahoo.fr; Rached.tourki@fsm.rnu.tn

***Abstract*-This paper presents different implementations of the LPC (Linear Predictive Coding) algorithm used in the majority of voice decoding standard. The windowing/autocorrelation bloc is implemented by three different versions on an FPGA Spartan3 with Microblaze soft core processor using Embedded Development Kit (EDK). Allowing the possibility to integrate the soft core processor a first solution consists of a pure software implementation of the LPC using this core RISC processor. Second solution is pure hardware architecture implemented using VHDL based methodology starting from description until integration. Finally, hardware/ software co-design is given. The hardware part consists of the autocorrelation core that solves the part of the algorithm with higher computational costs. The software part consists of the embedded soft core processor that manages the windowing bloc. The implementation results are presented, a comparison is then made among the three alternative architectures with different data lengths for performance and area.**

Keywords-Linear Predictive Coding; System on Programmable Chip; FPGA; Co-design

I. INTRODUCTION

Applications such as cell phones, hearing aids, and digital audio devices are applications with stringent constraints such as area, speed and power consumption. These complex applications can be well addressed by Systems On Programmable Chip (SoPC). Such applications require an implementation that meet these constraints with the minimum time to market.

Modern Field Programmable Gate Arrays (FPGAs) contain many resources that support DSP applications such as embedded multipliers, Multiply Accumulate (MAC) units and processor cores. The motivation for the introduction of such processor core comes from the idea that most FPGAs contained within an embedded system require interaction level with an external processor. Moving this processor onto the chip allows the FPGA and the processor to communicate without the bottlenecks associated to communicating with off-chip devices. Several programmable logic manufacturers offer FPGAs platform. Altera, Atmel, Xilinx[1] propose devices that integrate hardware cores of processors such as ARM, MIPS and PowerPC CPUs and/or allow instantiation of soft processor such as MicroBlaze from xilinx and Nios from Altera, DSP and microcontroller cores like PicoBlaze.

In this work, we propose different implementation modes of the Linear Predictive Coding (LPC) algorithm. According to its criticality for the LPC algorithm and its reusability property, the windowing/autocorrelation bloc seems a good candidate for IP design. This block is thus implemented by three different versions on an FPGA. First of all the candidate bloc has been programmed in C and executed using Microblaze core. Then it was implemented using VHDL description. Finally, the autocorrelation core is then proposed to be implemented using HW/SW architecture with the existing processor. Each architecture performances are compared for different data lengths.

The system performances have been evaluated with a prototyping board integrating embedded Microblaze processor with peripherals on a FPGA Spartan3. This flexible system enables to reduce overall product cost, achieve target performances, eliminate risk of obsolescence and finally use industry standard software development [2].

The remainder of this paper is organised as follows: Section 2 gives a necessary background on the LPC analysis and the autocorrelation algorithm. The methodology implementation is detailed in section 3. Each architecture implementation details are depicted in section 4. Section 5 presents the implementation results and finally section 6 concludes the paper.

II. LPC ANALYSIS OVERVIEW

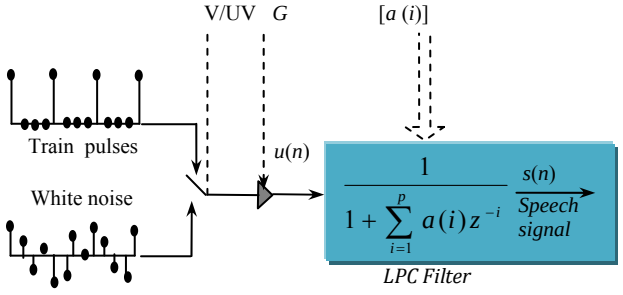
The LPC algorithm is a widely used technique for voice coder. In fact the human speech is produced in the vocal tract as a combination of vocal cords (in glottis) interacting with articulators. Actually, it can be modelled by the passage of an excitation signal through a filter. This model called linear predictive coding (LPC) is given in the case of an auto-regressive signal [3, 4] and is presented in figure 1a.

The correspondence between the real world and the mathematical models is given in figure 1.b[5, 6], This is equivalent to say that the input-output relationship of the filter is given by:

$$S(z)=G.\ u(z)/A(z)$$

In order to evaluate the LPC model, the linear regression has to be solved, that is to say the coefficients a(i) of the recursive filter are to be found. If the operations involved by the LPC analysis are summed-up, this task includes five sub-tasks:

- A windowing + an autocorrelation,
- The Levinson Durbin algorithm [7,8],
- A LP to LSP Conversion [9,10],
- A LSP Quantization [11],
- Interpolation and conversion of the LSP coefficients.



a)

Vocal tract	⇔	*H(Z)* (LPC Filter)
Air	⇔	u(n) (Innovation)
Vocal Cord	⇔	V (voiced)
Vocal Cord	⇔	T (pitch period)
Fricatives	⇔	UV (unvoiced)
Air Volume	⇔	u(n) (Innovation)

b)

Fig. 1. LPC Model

III. IMPLEMENTATION METHODOLOGY

In order to extract an efficient hardware/software partitioning, the processing and communication complexity of the LPC algorithm was evaluated in [12]. A survey of the distribution of the different processing complexity parts of the algorithm is necessary. The aim is to first know to what extent it can be interesting to achieve some blocks into hardware rather than software and, secondly, if this blocks can be used in other applications in order to justify their specification as a reusable component.

A. Profiling and Hardware/Software Partitions

The windowing/ autocorrelation block and the LSP Quantization block are the most critical blocks. The technique used for the quantization is specific to the standard G729. In fact the LSP coefficients are quantized using the LSF (line spectral frequency) representation in the normalized frequency domain (0, π). Then the quantization is achieved using a two-stage vector quantizer[13]. In fact, rather than doing the computations, most voice coders use a simple look-up table. This solution can be efficiently implemented into software. This is the way we have implemented this block. The other significant sub-block, the autocorrelation sub-block, is data computation dominated. We thus decided to centres round this block. In fact the critical operations of the majority of digital signal processing algorithms are usually the convolution or product accumulations. A dedicated arithmetic unit achieving a multiplication and accumulation is generally used to cope with this problem; it is called a MAC.

Where s(n) is the windowed speech signal used to compute the autocorrelation coefficients r(k). The LP analysis window is applied to 120 samples from the last speech frames, 80 samples from the present speech frame, and 40 samples from the future frame giving a total number of 240.

To achieve our objectives, we have used the Spartan™-3 Starter Kit provided by Xilinx[2].

B. Accelerating Embedded Applications

The Spartan-3 Starter Board from Xilinx (figure 2) provides a powerful, self-contained development platform for designs. It features a 200K gate Spartan-3, on-board I/O devices, and one MicroBlaze fast asyncronous SRAM, making it the perfect platform to experiment with any new design, from a simple logic circuit to an embedded processor core. The board also contains a Platform Flash JTAG-programmable ROM, so designs can easily be made non-volatile. The Spartan-3 Starter Board is fully compatible with all versions of the Xilinx ISE tools. It ships with a power supply and a programming cable, so designs can be implemented immediately with no hidden costs.

The Spartan 3 starter kit which combines XC3S200 of Spartan 3 FPGA family, provides an environment for designing an embedded and reconfigurable system based on the MicroBlaze softcore processor.

The MicroBlaze embedded soft core is a Reduced Instruction Set Computer (RISC) optimized for implementation in Xilinx FPGAs. With few exceptions, the MicroBlaze can issue a new instruction every cycle, maintaining single-cycle throughput under most circumstances.Unlike any other hard processors which are actually implemented in the FPGA at a transistor level, the soft core is an IP block written in HDL and is implemented in the free resources of an FPGA. Therefore it is configurable by choosing the components that would be included in the system.

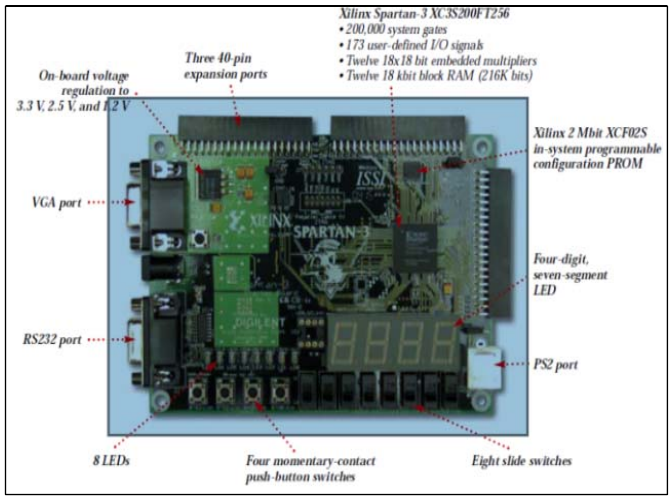


Fig. 2. Spartan-3 Starter Board

According to the application type, MicroBlaze processor can be configured in several ways to save power or area. MicroBlaze processor provides four bus connections, namely the Local Memory Bus (LMB), the On-chip Peripheral Bus (OPB), the Fast Simplex Link (FSL) and the Xilinx Cachelink (XCL). The LMB is a dedicated bus for MicroBlaze and on-chip block RAM connection. The OPB is a CoreConnect IBM standard bus [16] and gives the capability to connect variety of available IP blocks and peripherals.

The FSL has a FIFO-based interface and it provides a connection between a custom hardware to assist particular application. Lastly the XCL interface provides a link between MicroBlaze processor and data and instruction caches.

The implementation of our system will be carried out using XPS tool (Xilinx Platform Studio) provided by EDK [14] environment (Embedded Development Kits) as well as the Core Generator tools included in ISE (integrated software environment), to profit from Xilinx IP [15] respecting the constraint of time to market (TTM).

IV. SOLUTION SPACE EXPLORATION

The LPC block input is assumed to be a 16 bits PCM signal. Two pre-processing functions are applied before the encoding process signal scaling, and high-pass filtering. The scaling consists of dividing the input by a factor 2 to reduce the possibility of overflows in the fixed-point implementation. The high-pass filter serves as a precaution against undesired

low-frequency components. A second order pole/zero filter with a cut-off frequency of 140 Hz is used.

The Linear Prediction (LP) is performed once per speech frame using the autocorrelation method with a 30 ms asymmetric window. Every 80 samples (10 ms), the autocorrelation coefficients of windowed speech are computed and converted to the LP coefficients using the Levinson algorithm.

The LP analysis window consists of two parts: the first part is half a Hamming window and the second part is a quarter of a cosine function cycle. The window is given by:

$$W_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{399}\right) & n = 0,....199 \\ \cos\left(\dfrac{2\pi(n-200)}{159}\right) & n = 200,....,239 \end{cases}$$

The windowed speech is given by the following equation :

$$s'(n) = W_{lp}(n)s(n) \qquad n = 0,....,239$$

And is used to compute the autocorrelation coefficients as follow:

$$r(k) = \sum_{n=k}^{239} s(n)s(n-k) \quad k = 0....9 \begin{cases} r(0) = S(0).S(0) + ... + S(239).S(239) \\ r(1) = S(1).S(0) + ... + S(239).S(238) \\ ..... \\ ..... \\ ..... \\ r(9) = S(9)S(0) + ... + S(239).S(230) \end{cases}$$

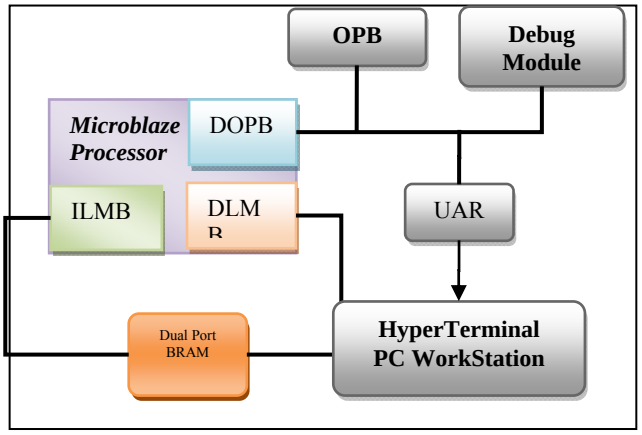


Fig. 3. Microblaze System Architecture

A. Software Implementation

In software version, LPC Algorithm is a full software implementation mapped onto the MicroBlaze processor. Data is accessed through the data cache and comes from the external SRAM.

The system architecture of a MicroBlaze implementation is shown in Figure 3. The LPC algorithm is re-coded in ANSI C and stored in on-chip BRAM. The size of BRAM is set to 64K which is the maximum size that one MicroBlaze processor can have on the Spartan-3 XC3S200 [2] FPGA. The connection between Block RAM and MicroBlaze processor is through ILMB[15] (Instruction-side Local Memory Bus) and DLMB[15] (Data-side Local Memory Bus). The DOPB[15] (Data-side Onchip Peripheral Bus) is used for connecting the following peripherals:

UART[15], OPB Timer[14], and Debug Module[15]. The HyperTerminal is a communication program used on a PC workstation to print the result received at PC's serial port. The Xilinx Microprocessor Debug (XMD) tool of Xilinx Embedded Development Kit (EDK) is used to debug the system from the PC workstation. The elapsed-time of running the C program in the Block RAM presented later is achieved by starting and stopping the OPB Timer peripheral before and after a specific code segment.

The timer count represents the number of cycles required to complete this operation. The profiling results of the two operations are shown in table 1. They are evaluated for several samples corresponding to several speech-coding.

Results in table 1 clearly indicate that the autocorrelation block utilizes 80 % of the total execution time. that's why it is selected to be implemented in hardware. Thus, main purpose is to improve these two components using HW/SW codesign.

TABLE I
PROFILING RESULTS WITH PURE SOFTWARE

	Execution time in µs					
Function	n=10	n=80	n=100	n=150	n=200	n=240
Windowing	16,67	133,33	166.67	250	333,33	400
Autocorrelation	533,33	2983	3683,3	5433,33	7183,33	8633,33

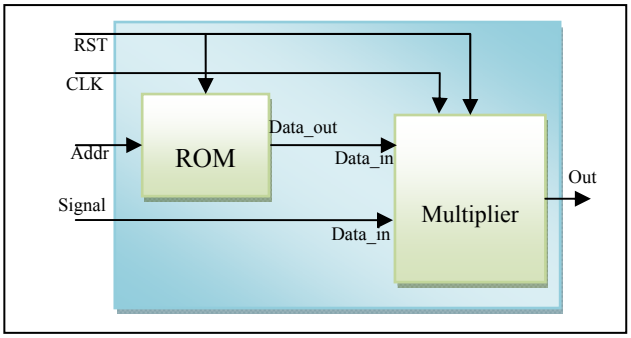


Fig. 4. Windowing Block Architecture

B. Hardware Implementation

The second architecture is a pure hardware implementation of the algorithm. This architecture makes use of an autocorrelation core that will be presented later in HW/SW architecture section. It also makes use of a windowing core in order to reduce the generation of side lobes in the frequency spectrum. To be done two blocks are implemented: multiplier block and a 256 word 16 bit ROM as shown in figure 4.

To minimize development time Xilinx's CORE generator has been used to produce the simple arithmetic operators and memory required by this windowing block.

The used ROM is constructed in the Spartan III FPGA using the ISE's CORE generator. In fact the CORE generator can be used to produce devices ranging in complexity from simple arithmetic operators and delay elements to complex building-blocks such as digital signal decoders, processing filters, multiplexers, transformers, FIFOs, and memories.

The content of this memory is defined by supplying an input coefficient precalculaed starting from the equation 2 .

The final system is built around the windowing core and the autocorrelation computation presented in section II.B. Figure 5 summarizes the global hardware system design.

This global design is implemented using the Spartan 3 Xilinx board. The RTL specification is made compatible with most of FPGA design tools for an efficient component reuse. This is also the reason why we have used hand-written RTL MAC rather than on-chip one.

The specification can be parameterized according to the window size and the data width (word length). It is also parameterized according to the value of index k of the r(k) autocorrelation coefficient equation.

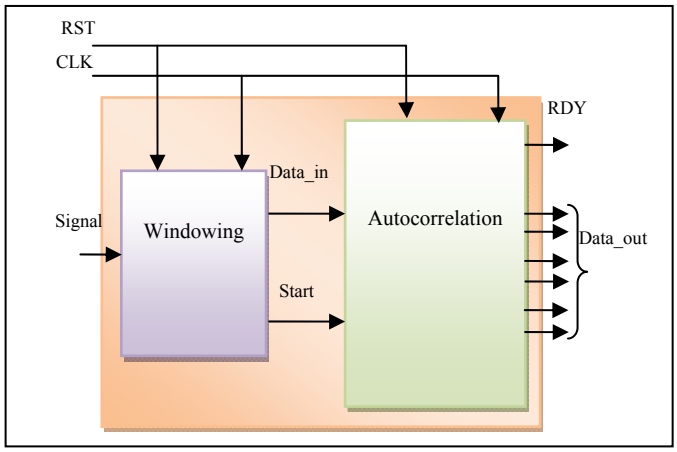


Fig. 5. Global Hardware Architecture

TABLE II
PROFILING RESULTS WITH PURE HARDWARE IMPLEMENTATION

	Execution time in µs					
Function	n=10	n=80	n=100	n=150	n=200	n=240
Windowing	0,121	0,811	1,01	1,509	2,001	2,406
Autocorrelation	0,231	1,539	1,919	2,867	3,802	4,572

The profiling results of the two operations are summarized in table 2.

C. *Hardware/Software Implementation*

In this version, we proposed a HW/SW high-performance implementation. As shown in software version, profiling algorithm shows that the autocorrelation could be implemented into hardware block. Thus we have applied an EDK based design flow for the implementation of the windowing/autocorrelation algorithm. In fact The EDK is an integrated environment for HW & SW co-design.

One of the biggest challenges of this architecture was to get a System On a Programmable Chip (SOPC) with LPC algorithm. This means implementing both software and hardware components. As the target device, a Spartan-3 starter kit was chosen due to its flexibility, great promise of integrating both the hardware and software co-designs into one flow as shown in fig. 6.

The soft core Processor MicroBlaze is used in a stand-alone mode to run a software program (written in C) which is loaded into BRAM.

The whole design flow, except the simulation execution, is covered by Xilinx EDK and ISE software packages. The Adding a Processor System to and FPGA Design module introduces the two design flows of a hardware system available in EDK: the XPS flow and the ISE flow. The procedure involved in each flow is illustrated figure 6. This flow summarizes our system design. The speech signal is stored in the SRAM. The MicroBlaze processor achieves the windowing computation then sends the results to the Autocorrelation hard Block to compute the autocorrelation coefficients.

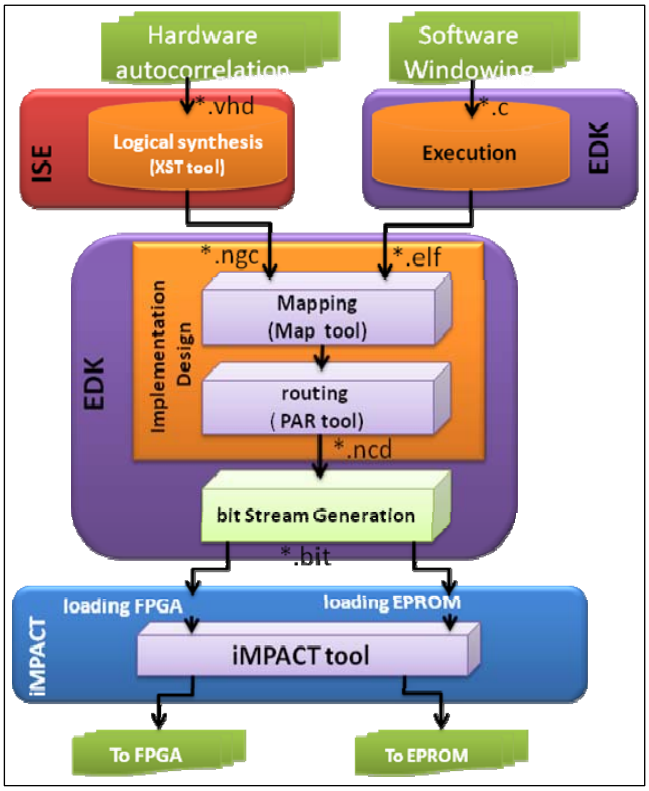


Fig. 6. Embedded Development Tool Flow Overview

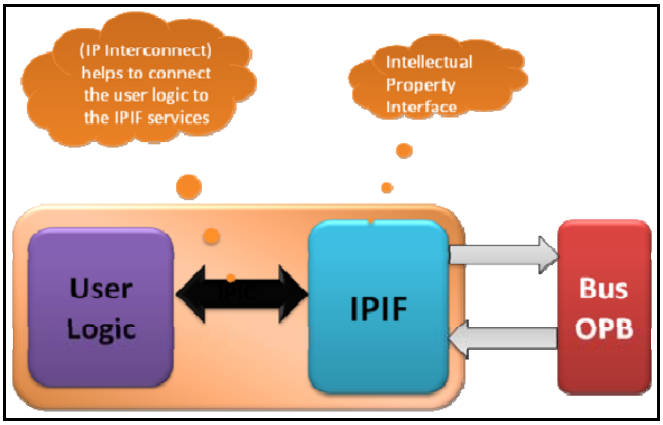


Fig. 7. OPB Integration

In this architecture the user logic is an Autocorrelation core which is coded in VHDL. It should be noted that linear prediction in speech processing has an important characteristic since it is determinist. Taking this characteristic into account it is usually possible to apply some parallelism techniques in order to increase the performance of the implementation.

The Microblaze achieves the windowing computation, then sends the result to the user logic The FPGA computes the autocorrelation coefficients.

In our case the autocorrelation computation can be split in many sub-tasks independently executable since the set of computations can be written as in equation (4)

Systolic arrays are very well suited to compute this kind of computation[17]. Applying the dependence method [18], a linear systolic array (figure 8) has been specifically designed for the autocorrelation computation with 10 MAC-based cells. For a clock cycle i, every MAC(k) reads S(i) and S(i-k) and adds to its previous result S(i)*S(i-k). At the first clock edge MAC0 performs the multiplication S(0).S(0). At the second one, MAC0 performs S(0).S(0)+ S(1).S(1), and MAC1 performs the multiplication S(1).S(0). At the third clock edge MAC0 performs S(0).S(0)+S(1).S(1)+S(2).S(2), MAC1 S(1).S(0) + S(2).S(1) and MAC2 S(2).S(0) , and so on.

Finally, the ten autocorrelation coefficients r(0)…r(9) are provided after 240 clock periods.

This linear array achieves an efficient speed-area trade off with a parallelism rate of 10 (10 useful multiplications and accumulations each clock cycle on average). A single clock is used to control this array. Furthermore, since this systolic array is linear (one dimensional array), the data communication interface is also easy to be (re)used.

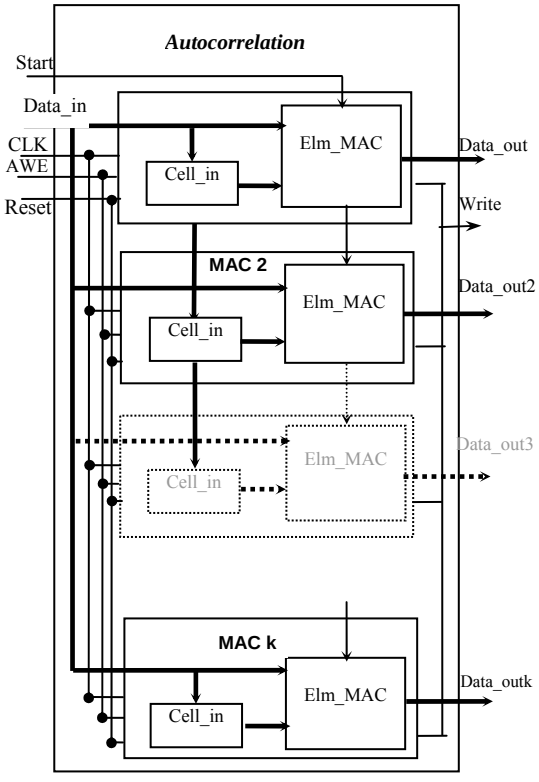


Fig. 8. Linear Systolic Array for the Autocorrelation

Systolic arrays are very interesting for reusable components. This kind of array uses elementary cells locally interconnected and is basically regular. That allows the design of soft IPs (according to the VSIA taxonomy [19]) with a generic parameter representing the number of elementary cells. In our case, this parameter corresponds to the index k of the autocorrelation coefficient equation r(k).

The size of the proposed array (number of MAC cells) does not depend on the maximum value of the n index : this value only sets the number of input data, i.e. it sets the size of the hamming window. The proposed array can thus be easily used in many speech coders (GSM 160, G723, G729 240, etc.) [13, 20]. Another interesting point of this autocorrelation implementation is that it doesn't need intermediate data to be stored in a RAM. Additional memory accesses and control are thus avoided.

The profiling of the HW/SW architecture is shown in table 3. It can be seen that the gain in execution time is considerable as soon as the number of samples increases. In fact the autocorrelation core reduces the execution time by a ratio close to 96%.

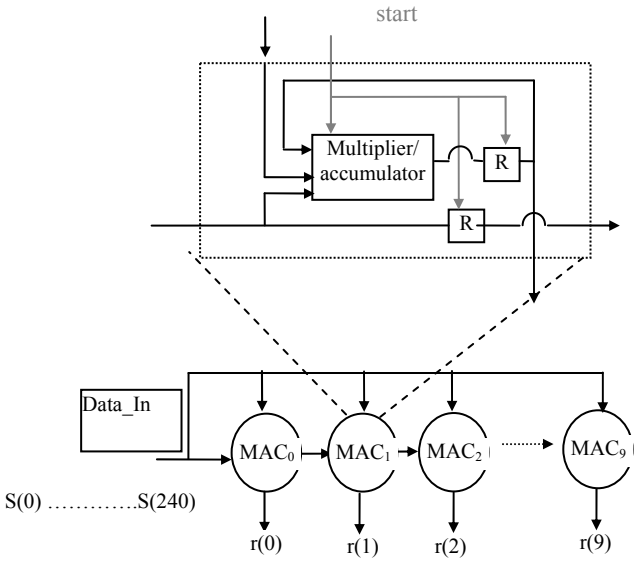


Fig. 9. Hardware MAC Architecture

TABLE III
PROFILING RESULTS AFTER USING HARDWARE ACCELERATOR

	Execution Time in µs					
Function	n=10	n=80	n=100	n=150	n=200	n=240
Windowing (SW)	16,67	133,33	166.66	250	333,33	400
Autocorrelation(HW)	0.231	1,539	1,919	2,866	3,801	4,572

V. RESULTS AND DISCUSSION

In this paper, three different architectures were proposed for realizing a windowing and autocorrelation computation. These three architectures are implemented on Spartan-3 board.

Using a window sized of 240 and a word lenth of 16bits, the profiling results for each architecture are shown in figure 11. It is clear that the pure hardware implementation results in a significant speedup over the software implementation and rather HW/SW version.

The hardware resources for the implementation are summarized in figure 12. The SW architecture resource requirements are fixed with the value of n all versions of the algorithme are implemented in software and changing n requires just modifying the C code and recompiling it. We note that the hardware version, use more silicon surface then HW/SW version and then over consumption.

The results show that the increased hardware resource requirements are not excessive when compared to the basic MicroBlaze core.

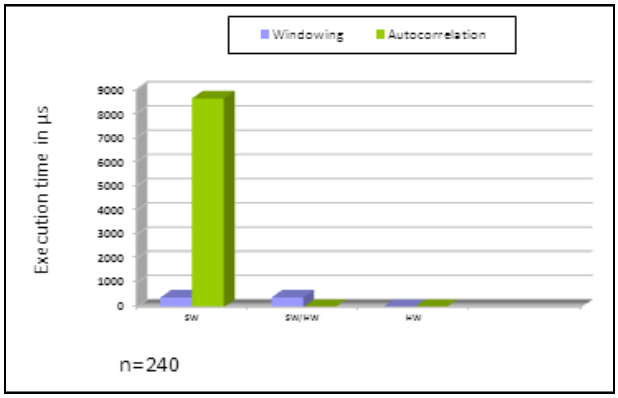


Fig. 11. Profiling Results for Different Architectures

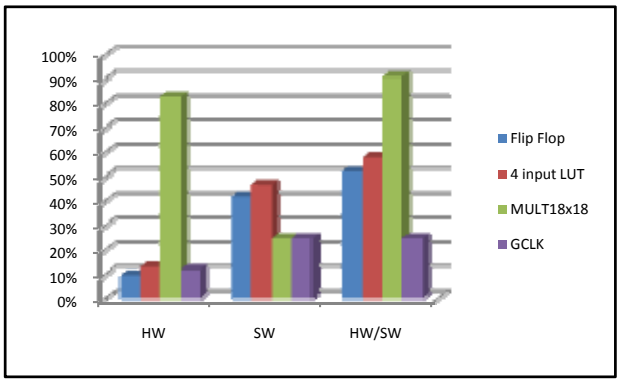


Fig. 12. Resource Utilization for Different Architectures

VI. CONCLUSION

In this paper three different architectures were proposed to implement the LPC algorithm. All of them are aimed for voice decoding process using the Xilinx board based MicoBlaze soft processor. A comparison between the three architectures shows that using a hardware architecture coupled with a MicroBlaze processor in mixed architecture reduces the number of cycles required to perform the most critical operation by about 96%. Also, the HW/SW approach allows the designer to focus only on the development of the VHDL core description without taking into account communication problems. This goal is reached due to the definition of automated flow IP-Cores integration into complete system architecture without requiring the user interaction. Using a HW/SW architecture gives a significant speed up results with a moderate area and lower flexibility. Thus, such a design will enable us to achieve an optimum tradeoff between speed and flexibility permitting to make a complete system on programmable chip (SoPC). In fact the proposed approach represents a general computing model which can be extended to many different applications like G729 coder.

REFERENCES

[1] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays," Springer-Verlag, Berlin, Germany, 2nd edition, 2004.

[2] Spartan-3 FPGA Starter Kit Board User Guide, UG130 (v1.2), (June 20, 2008).http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf

[3] H. M. Zhang and P. Duhamel, "Doubling Levinson/Schur Algorithmand its Implementation," Proceedings International Conference on Acoustics, Speech, and Signal Processing ICASSP, Glasgow, UK, 23-26 May 1989, pp. 1115 - 1118.

[4] B. S. Atal and M.R. Schroder, "Linear Prediction Analysis of Speech Based on Pole Zero Representation," Journal of Acoustical Society of America, Vol. 64, No. 5, 1978, pp. 1310–1328.

[5] L. R. Rabiner and R. W. Schafer, "Digital Processing of Speech Signals", Prentice Hall Edition, September 1978.

[6] B. Gold and N. Morgan, "Speech and Audio Signal Processing: processing and perception of speech and music," Wiley edition, 2000.

[7] P. Delsarte and Y. Genin, "The Split Levinson Algorithm,". IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 34, No. 3, 1986, pp.470–478.

[8] R. Hagen and P. Hedelin, "Spectral Coding by LSP Frequencies- Scalar and Segmented VQ-Methods," IEE Proceedings-I, Communications, Speech and Vision, Vol. 139, No.2, 1992, pp. 118–122.

[9] N. Sugamora and F. Itakura, "Speech Analysis and Synthesis Methods Developed at ECL in NTT- From LPC to LSP," Speech Communication, Vol.5, No. 2, 1986, pp. 199–215.

[10] R. Salami, C. Laflamme, J-P. Adoul and A. Kataoka, "Design and Description of CS-ACELP: a toll Quality 8 Kb/s Speech Coder," IEEE Transactions on Speech and Audio Processing, Vol.6, No. 2, 1998, pp. 116-130.

[11] P. Zador, "Asymptotic Quantization Error of Continuous Signals and the Quantization Dimension", IEEE Transactions on Information Theory, Vol.28, No.2, 1982, pp. 139–148.

[12] F. Sayadi, E. Casseau, M. Atri, M. Marzougui, R. Tourki and E. Martin, "G729 Voice Decoder Design," The Journal of VLSI Signal Processing, Springer, Vol.42, No.2, 2006, pp. 173-184.

[13] Coding of speech at 8 kbit/s using conjugate structure algebraic code excited linear prediction (CS-ACELP), ITU-T Recommandation G729 (03/96). http://www.ece.cmu.edu/~ece796/documents/g729.pdf

[14] EDK Concepts, Tools & Techniques, XTP013 EDK (10.1 2008). http://www.xilinx.com/support/documentation/sw_manuals/edk10_ctt.pdf

[15] Microblaze Processor Reference Guide (01/17/08). http:// www. xilinx.com/ support/documentation/sw_manuals/mb_ref_guide.pdf

[16] M. Maaref, "Creating an OPB IPIF-based IP and Using it in EDK". Xilinx, XAPP967 (v1.1), 2007. http:// www. xilinx.com /support /documentation/application_notes/xapp967.pdf

[17] E. Casseau and D. Degrugillier, "Linear Systolic Array for LU decomposition". Proceedings of the 7th International Conference on VLSI Design, Calcutta, India, 1994, pp. 353–358.

[18] C. Tayou, P. Quinton, S. V Rajopadhye and T. Risset, "Derivation of Systolic Algorithms for the Algebraic Path Problem by Recurrence Transformations," Parallel Computing,Vol. 26, No.11, 2000, pp. 1429–1445.

[19] Virtual Socket - VSIA Alliance - http://www.vsia.org.

[20] A. Spanias, T. Painter and V. Atti, "Audio Signal Processing and Coding," Wiley edition, March 2007.